CERN-TH.7420/94

PROJET:
Jet Cross Sections in Deeply Inelastic Electron Proton Scattering
Version 4.1

Dirk Graudenz [1]
Theoretical Physics Division, CERN
CH–1211 Geneva 23

Abstract

PROJET is a parton level Monte Carlo program for the calculation of jet cross sections in deeply inelastic electron proton scattering. In its present version it contains the Born level diagrams for the production of (1+1), (2+1) and (3+1) jets and the next-to-leading order corrections for the production cross sections of (1+1) and (2+1) jets for all polarisations of the exchanged virtual photon. In particular, the full angular correlations between the lepton and jet momenta are implemented. The program permits the application of acceptance cuts on all external momenta. For this purpose, the program creates an event record accessible to the user program with all momenta in the laboratory frame and in the center of momentum frame of the proton and the virtual photon. This option is indispensable for phenomenological studies because of the strong dependence of cross sections on phase space restrictions and the large uncertainty of fragmentation corrections in the proton direction. Since PROJET uses the Monte Carlo integration method for the evaluation of phase space integrals, the weights of the generated events can be used to produce distributions of observables related to jet momenta.

CERN-TH.7420/94
August 1994

[1] *Electronic mail addresses: graudenz @ cernvm.cern.ch, I02GAU @ DHHDESY3.bitnet*

PROGRAM SUMMARY

Title of the program: `PROJET` version 4.1.

Computer: Any computer with `FORTRAN 77` compiler.

Programming language used: `FORTRAN 77`.

High speed storage required: Size of executable program is approximately 2.5 MBytes.

No. of cards in combined program and testdeck: about 40.000.

Other programs used: `VEGAS` [Lep78, Lep80] (multidimensional Monte Carlo integration routine), `PAKPDF` [Cha92], `PDFLIB` [Plo93] (parametrizations of parton densities).

Keywords: Quantum Chromodynamics (QCD), Jet Physics, Deeply Inelastic Electron Proton Scattering, Monte Carlo Simulation.

Nature of physical problem: In deeply inelastic scattering, the hadronic final state can be analysed by a jet cluster algorithm resulting in cross sections for the production of (n+1) jets (including the target remnant jet). It is possible to extract physical parameters (Λ_{QCD}, parton densities $f_i\left(\xi, \mu_f^2\right)$) from experimentally measured jet cross sections.

Method of solution: `PROJET` makes it possible to study these cross sections from a theoretical point of view by performing the integration of the differential cross sections over phase space regions that are related to a certain number of jets. QCD corrections are included by analytical formulae.

Restrictions on the complexity of the problem: The program contains jet cross sections for the production of (1+1), (2+1) and (3+1) jets. QCD corrections to (1+1) and (2+1) jet cross sections are included for all polarisations of the exchanged virtual photon.

Typical running time: Depends strongly on the process under consideration; the generation of the weight for 1 event takes typically 0.001 to 0.02 seconds on a DEC or hp workstation, depending on the process; therefore an integration using 200.000 events typically needs 3 minutes to 1 hour of CPU time.

1 Introduction

Recent results from HERA show evidence for events with a pronounced jet-like structure in the hadronic final state [HERA]. In order to perform quantitative tests of perturbative QCD, next-to-leading (NLO) order predictions have to be confronted with experimental data. In principle, there are two basic tools to study jet physics from a theoretical point of view:

- Event generators
- Parton level Monte Carlo programs.

Programs of the first class (event generators) simulate the underlying physics by generating full events with a hadronic final state with probabilites given by some hard processes based on exact matrix elements, a leading logarithmic correction using parton showers and a final hadronisation step based on a specific model motivated by phenomenology. A certain set of parameters can be adjusted to fit experimental data very precisely. Event generators are a very convenient tool to make detector studies. Examples are `ARIADNE`, `HERWIG`, `LEPTO` and `PYTHIA` [HERA91].

The second class of programs (parton level Monte Carlo programs) is based on a different approach. Their goal is to make predictions using only a very small number of parameters (basically $\Lambda_{\overline{\rm MS}}$ and a certain set of parton densities $f_i\left(\xi,\mu_f^2\right)$). The final state consists of partons clustered by a means of a suitable jet algorithm and integrated over singular phase space regions. Therefore assumptions have to be made on how to map the results to a realistic experimental situation. The correspondence between the parton level and the hadron level is made by a suitable jet definition (in principle it is assumed that a hard parton accompanied by collinear and soft partons can be identified with a hadronic jet). Jet definitions in deeply inelastic scattering based on a cluster algorithm are described elsewhere [BKMS89, GM92, Gra90, Gra91, Gra94a, KMS89], so no further comments are made on this problem here. Because of the small number of parameters that are involved in parton level Monte Carlo programs it it possible to extract fundamental parameters of the underlying field theory ($\Lambda_{\overline{\rm MS}}$and possibly the parton densities $f_i\left(\xi,\mu_f^2\right)$).

`PROJET` is a parton level Monte Carlo program for the calculation of jet cross sections in deeply inelastic electron proton scattering on the basis of Born terms and NLO corrections. Recently, another program (`DISJET`) for jet cross section calculations has been published [BM94]. It can be used to calculate the total cross section in NLO for (1+1) and (2+1) jet production. For this purpose, the metric and longitudinal polarisations of the exchanged virtual photon as implemented in `DISJET` are sufficient[2]. In the case of cuts on the jet momenta this statement is no longer true and a full implementation of the matrix elements is indispensable. Moreover, jet cross sections strongly depend on restrictions of the phase space of outgoing particles. These restrictions appear naturally in experimental situations because of acceptance cuts for detectors (an example being given by angle cuts in the case of the beam pipe at the eP collider HERA) and possibly theoretically in order to restrict calculations to phase space regions where the use of fixed order perturbation theory is justified. In addition, fragmentation corrections are not well known in the case of jets in the forward direction[3]. In order to enable studies of the kind just mentioned, `PROJET` supplies the user with an event record for every generated event, thus enabling him to restrict the phase space arbitrarily. In addition, since the weight of every generated event is known as well, distributions and expectation values of observables depending on external momenta can be studied. A detailed phenomenological study is in preparation [Gra94b].

It should be stressed that `PROJET` is not intended to be a tool that works like an event generator. First of all, the program does not produce events with unit weight, but simply uses the Monte Carlo

[2]For a discussion, see Section 2.

[3]The various QCD inspired models predict different shapes of the transverse energy flow in the forward direction, for an overview see [Fel94]. The forward direction is defined to be the direction of the incoming proton.

method to do a multidimensional integration, and secondly, the final state consists of jets and not of hadrons. Therefore the output of the program can only be compared to experimental results if the latter are corrected for detector acceptance and processed by the same jet cluster algorithm (mJADE in the present implementation, see [GM92]) used in the program. This means in particular that the jet cut c has to be the analysis cut of the experimental cluster algorithm. Generating `PROJET` events with a very small cut and performing a subsequent clustering is *not* a viable procedure from the theoretical point of view since this does not take QCD corrections into account in the correct way.

This manual is organised as follows. In the next section the matrix elements used in `PROJET` are discussed. In Section 3 the general structure of `PROJET` is described. The program calculates the four-vectors of the external particles and stores their values in an event record. The information contained in the event record is described in Section 4. A user interface is explained in Section 5. Finally a complete list of the program parameters that are implemented and a list of possible error messages of the program is given in Section 6.

2 Matrix Elements

`PROJET` integrates matrix elements for the processes

$$\mathrm{e}^- + \text{proton} \rightarrow \text{proton remnant} + \text{n jets} \tag{1}$$

for n up to 3. Since the proton remnant is counted as a jet, matrix elements for (1+1), (2+1) and (3+1) jet production are implemented.

The cross section in the (2+1) jet case can be decomposed as a sum over helicity cross sections

$$\begin{aligned}\sigma &= \frac{1+(1-y)^2}{2\,y^2}\,\sigma_M + \frac{1+(1-y)^2+4(1-y)}{2\,y^2}\,\sigma_L \\ &+ \frac{2-y}{y^2}\sqrt{1-y}\,\cos\Phi\;\sigma_\Phi + 2\frac{1-y}{y^2}\,\cos 2\Phi\;\sigma_{2\Phi}\end{aligned} \tag{2}$$

including those which are mediated by the exchange of a virtual photon with metric and longitudinal polarisation (σ_M and σ_L, respectively)[4] (a detailed discussion can be found in [KMS89]). Here and in the following the relevant kinematical variables are

$$\begin{aligned}&x_B = \frac{Q^2}{2Pq}, \quad y = \frac{Pq}{Pk}, \\ &S_H = (P+k)^2, \quad Q^2 = -q^2 = S_H\,x_B\,y, \quad W^2 = S_H\,(1-x_B)\,y,\end{aligned} \tag{3}$$

where P is the proton momentum, q is the momentum of the exchanged virtual photon, and k is the momentum of the incoming lepton. Φ is the angle of the plane defined by the incoming and outgoing leptons and that of one outgoing jet and the incoming proton, respectiveley, in the center of momentum (CM) frame of the incoming proton and the virtual photon. Integrated over Φ, the total cross section can be expressed in terms of the metric and longitudinal cross sections (σ_M and σ_L) alone (where in many cases the metric contributions dominate). This statement is no longer true if one imposes angular cuts on the outgoing particles, because then the integration domain is restricted, and the additional terms do not integrate to zero any longer.

[4] Metric contributions are defined by the contraction of the hadronic tensor $H_{\mu\nu}$ with the metric $-g^{\mu\nu}$, and longitudinal contributions by the contraction with $p_0^\mu p_0^\nu$, where p_0 is the momentum of the incoming parton. Projection operators for the other helicity cross sections can be defined as well.

For the Born level, the relevant Feynman diagrams and references to the literature can be found in [BKMS89, GM92, KMS89]. The Born matrix elements have been implemented in two different ways. One implementation (`NPARTONS` >0, `NPARTONS` is a parameter which is explained later on) uses the contraction of the complete lepton tensor with the hadron tensor, the other implementation (`NPARTONS` <0) uses the contributions from the explicit projections on the various polarisations of the virtual photon. Therefore, longitudinal contributions can be studied separately.

The NLO contributions in `PROJET` for the production of (2+1) jets for the metric polarisation of the virtual photon are those from [Gra90, Gra91, Gra94a][5]. The NLO corrections for the other polarisations of the exchanged virtual boson can be found in [BK91]. However, they have been recalculated for the present implementation in order to have a form of the analytical formulas that could be easily implemented in one of the recent versions of `PROJET`. These matrix elements are correct up to terms $\mathcal{O}(c \log^2 c)$, where c is the jet cut. In addition, the NLO contribution for the production of (1+1) jets (metric and longitudinal polarisation of the virtual photon) is implemented by subtracting the analytically integrated (2+1) jet cross section of the order $\mathcal{O}(\alpha_s)$ from the total cross section of $\mathcal{O}(\alpha_s)$ (see, for example, [AEM79, KMS89, Gra94a]).

A complete list of cross sections implemented in `PROJET` can be found in table 1 (see Section 6).

3 Program Structure

`PROJET` consists of a set of subroutines. The user has to supply a main program that controls `PROJET` by calls to the subroutine `setpar(ipar, value, ierr)`. `ipar` specifies whether a parameter of `PROJET` has to be changed or a specific action should be performed. `value` is the new value of the selected parameters (double precision), and `ierr` is an error code. An error condition is given by `ierr` $\neq 0$. The possible values of ipar are given in the following list:

- `ipar=(-3)`:
  Initialize `PROJET` and preset all parameters.
- `ipar=(-2)`:
  Initialize the adaptive integration routine `VEGAS`.
- `ipar=(-1)`:
  Initialize phase space integration. Start integration.
- `ipar=0`:
  No operation ("comment").
- `ipar>=1`:
  Set a `PROJET` parameter. See Section 6.

If a reference to a parameter could not be resolved (i.e., if `PROJET` does not know this parameter), the user supplied subroutine `userset` is called.

The general structure of the main program should be the following:

- (a) initialize `PROJET` and `VEGAS` by subsequent calls `call setpar(-3, 0.d0, ierr)`, `call setpar(-2, 0.d0, ierr)`.
- (b) modify the default parameters of `PROJET` by calls `call setpar(ipar, value, ierr)`.

[5] Please note that there is a sign error in the virtual corrections in [Gra90, BK91]. This has been corrected in [Gra94a]. All `PROJET` versions since early 1993 use the correct matrix elements.

- (c) start the integration by `call setpar(-1, 0.d0, ierr)`.
- (d) if necessary, repeat steps (b) and (c).

The user has to supply two subroutines `user`, `userset` and a function `iusercut`. They are explained in Section 5.

There is an example program `MCEX.F` reading a parameter file `PAREX` that calculates various jet cross sections. This program may be used as a guideline for main programs. To facilitate a check of the program, the file `RESEX` contains the output from file `unit=6`, and the file `PLOTEX` the output from file `unit=55`. In this particular test run program, the numbers should not be taken literally, since the integration uses only a small number of points in order to facilitate running the program, and results in general differ because the floating point accuracy depends on the machine where the program is running.

`PROJET` prints status information on the standard output file (`unit=6`). There is no input required from the standard input file (`unit=5`). All floating point variables in `PROJET` are `double precision` variables. The file units `unit=55,63,64,65,91` are reserved as output files for `PROJET`.

`PROJET` uses the following programs:

- The phase space integrations in `PROJET` are performed by the adaptive integration routine `VEGAS` [Lep78, Lep80] (for (3+1) jets the total phase space is 8-dimensional).
- The parton density parametrisations `PAKPDF` are from [Cha92]. In addition, an interface to the `PDFLIB` library [Plo93] of parton densities is implemented.

In its present form `PROJET` contains code that was used for other purposes (interface to an event generator, histograms, ...). These parts will be removed in the future when they are no longer needed and should therefore not be used.

4 The Event Record

If a generated event fulfills all cuts its momentum four-vectors in the laboratory frame (LAB frame) and in the CM frame of the proton and the virtual photon (PVP frame) are stored in the event record. The corresponding variables can be found in the common blocks `/evtrecord/` and `/pvprecord/`. Additional information is stored in `/userwgt/`:

```
      integer nmaxent
      parameter (nmaxent=20)
      double precision pout,ppol
      integer nentry,iptype,iident,inout
      common /evtrecord/
     &       pout(nmaxent,4),
     &       ppol(nmaxent,4),
     &       nentry,iptype(nmaxent),iident(nmaxent),inout(nmaxent)

      double precision pvp,pvppol
      common /pvprecord/
     &       pvp(nmaxent,4),
     &       pvppol(nmaxent,4)
```

```
      double precision evwgt,evxsect,everror
      integer ievaccpt
      common /userwgt/
     &       evwgt,
     &       evxsect,everror,
     &       ievaccpt
```

The user has access to these common blocks in the user routine `user` and in the function `iusercut`. The number of entries in the event record is given by `nentry`. For every entry `i` ($1 \leq$ `i` $\leq$ `nentry`), `pout(i,1..3)` is the 3-momentum of the particle in the LAB frame (1 is the x-direction which is always in the plane of the incoming electron and the outgoing electron, 2 is the y-direction and 3 is the positive z-direction which is defined to be the direction of the incoming proton; if the proton is at rest (`EPEE1` < 0, see Section 6), then the z-direction is defined to be the direction opposite to the incoming electron) and `pout(i,4)` is its energy (in units of GeV). Furthermore, the polar coordinates in the LAB frame are stored in the array `ppol`, namely, `ppol(i,1)` is the energy, `ppol(i,2)` is the modulus of the momentum, `ppol(i,3)` is the polar angle in radians, and `ppol(i,4)` is the azimuthal angle in radians. The corresponding four momenta in the PVP frame are stored in the variables `pvp` and `pvppol`, respectively.

The momenta of the following particles are stored in the event record:

- `iident(i)=(-4)`: incoming proton,
- `iident(i)=(-3)`: incoming electron,
- `iident(i)=(-2)`: outgoing electron,
- `iident(i)=(-1)`: exchanged virtual photon,
- `iident(i)=0`: remnant jet,
- `iident(i)=1,2,...`:parton jets.

The array `iident` allows the identification of the entries in the event record. In addition, there is an array `iptype` that specifies the particle type:

- `iptype(i)=0`: unspecified particle,
- `iptype(i)=1`: electron,
- `iptype(i)=2`: jet,
- `iptype(i)=3`: proton,
- `iptype(i)=4`: virtual photon.

Finally, there is an array `inout` specifying whether a particle in the event record is incoming, internal, or outgoing:

- `inout(i)=(-1)`: incoming,
- `inout(i)=0`: internal,
- `inout(i)=1`: outgoing.

The following table is an example for an event record of a (3+1) jet event with $E_P = 820$ GeV, $E_e = 26.7$ GeV. The first column gives the index of the entry, the second column is `iptype(i)`, the third column is `iident(i)`, the fourth column is `inout(i)`, and the other columns are the `pout(i,j)` in the order E, p_x, p_y, p_z.

```
1   3  -4  -1    0.82000D+03    0.00000D+00    0.00000D+00    0.82000D+03
2   1  -3  -1    0.26700D+02    0.00000D+00    0.00000D+00   -0.26700D+02
3   1  -2   1    0.27809D+02    0.11892D+02    0.00000D+00   -0.25138D+02
4   4  -1   0   -0.11096D+01   -0.11892D+02    0.00000D+00   -0.15613D+01
5   2   0   1    0.66116D+03    0.00000D+00    0.00000D+00    0.66116D+03
6   2   1   1    0.71842D+02   -0.53358D+01   -0.10878D+00    0.71643D+02
7   2   2   1    0.21690D+02   -0.15676D+01   -0.24633D+00    0.21632D+02
8   2   3   1    0.64201D+02   -0.49885D+01    0.35510D+00    0.64005D+02
```

To obtain the index `i` of a particle identification `iident1`, there is a function `ievid(iident1)` that returns the index `i`. The following example stores the momentum four-vector of the outgoing electron in the variables `p0`, `p1`, `p2`, `p3`:

```
c       identification of the outgoing electron is -2
        iident1=-2
c       get index
        i=ievid(iident1)
c       get four-vector
        p0=pout(i,4)
        p1=pout(i,1)
        p2=pout(i,2)
        p3=pout(i,3)
```

The relative weight of every event is given by the variable `evwgt`. The sum of `evwgt` of all events in the last integration of `VEGAS` is the cross section for the process in units of [pb]. Finally, there is a flag `ievaccpt` showing whether an event passed all cuts and the event has been generated in the final `VEGAS` integration (`ievaccpt=1`) or the event has been rejected or has been generated in one of the grid definition iterations of `VEGAS` (`ievaccpt=0`).

After the integration has been performed, the integrated cross section is stored in `evxsect`, and the estimated error of the integration in `everror` (this error estimate by `VEGAS` is not always reliable, see [Lep78, Lep80]).

5 User Defined Subroutines

The user has to define subroutines

```
        subroutine user(iopt,ierr)
        integer iopt,ierr

        subroutine userset(ipar,idata,xdata,ierr)
        integer ipar,idata,ierr
        double precision xdata
```

and a function

```
        function iusercut
        integer iusercut
```

of integer type.

`user` is called at different stages of the program:

- `iopt=1`: Called after the preset of all parameters (by a call of `setpar(-3,...)`). Here the user can modify values of the presets for all parameters and open files used for output purposes.
- `iopt=2`: Called after the initialisation of the integration routine (by a call of `setpar(-2,...)`).
- `iopt=3`: Called before the start of the integration (by a call of `setpar(-1,...)`). At this point the user should initialize histograms and variables used for the averaging of observables.
- `iopt=4`: Called in the case of every event. Here the user can bin observables into histograms or update averages. This subroutine is called even if an event has been rejected (if it did not satisfy any of the cuts). The variable `evwgt` contains the weight of the event (summing to the total cross section in [pb]). Here the flag `ievaccpt` can be tested. `user(4,...)` is called also during the grid definition runs of `VEGAS`.
- `iopt=5`: Called before the program terminates. This is the place to write histogram files to an external file or to print results.
- `iopt=6`: Called in case of an error. If something went wrong and the program did not crash before, `ierr` contains an error code. If `ierr` < 0, the error is fatal and `PROJET` will stop. If `ierr` > 0, the error is not fatal, and `PROJET` will continue.

`userset` is called if a parameter is not known to `PROJET`. `ipar` is the number of the requested parameter, `idata` is its integer value and `xdata` its double precision value. `userset` must return an error code in `ierr`. If the error code is not zero, `PROJET` will issue an error message. `userset` will be called once after `setpar(-3,...)` with `ipar=-3` to give the user the opportunity to initialize his own parameters.

`iusercut` is called after the kinematical variables have been calculated, but before the cross section is evaluated. The user can inspect the event record and impose additional cuts. If an event is to be rejected, `iusercut` should return 0, else 1.

6 Program Parameters

In this section the input parameters that can be used with `setpar` are described in detail.

In the following list `(VARIABLENAME)` gives the name of the variable that is affected. It is not necessarily the case that `par` is assigned to the variable, sometimes a function of `par` is assigned. `(FLOAT)` means that a parameter takes floating point values, and `(INT)` means that it takes integer values. `(DEF=XXX)` indicates that the default if no input is given is `XXX`. Parameter options marked with ($\bowtie$) have been implemented due to requests from `PROJET` users. They are included here for completeness only. It is recommended to consult the program author or to study the source code of `PROJET` before use is made of one of these parameters because they are sometimes restricted in use.

- `ipar=2001:` `(KPAR)` `(INT)` `(DEF=4)`
  Specifies which lepton variables are fixed and which are integration variables.
  0: x_B and y fixed;
  2: x_B and Q fixed;
  4: integrate over x_B and Q^2 within the cuts #2030–#2037.
  If x_B and y or Q are fixed (`KPAR=0,2`), the program returns $d\sigma/(dx_B dy)$ in [pb], if `KPAR=4`, the integrated cross section in [pb] is returned.
- `ipar=2002:` `(SH)` `(FLOAT)` `(DEF=295.93)`
  Specifies the CM-energy $\sqrt{S_H}$ of the incoming electron and proton.

- ipar=2003: (XH) (FLOAT) (DEF=0.1)
  Specifies Bjorken-x x_B in the case of KPAR=0, 2.

- ipar=2004: (Y) (FLOAT) (DEF=0.2)
  Specifies the lepton variable y in the case of KPAR=0.

- ipar=2007: (QQ2) (FLOAT) (DEF=100.0)
  Specifies the momentum transfer of the photon $\sqrt{Q^2}$ in the case of KPAR=2.

- ipar=2008: (CUT) (FLOAT) (DEF=0.01)
  Specifies the invariant jet cut c. The cut condition depends on #2046.

- ipar=2014: (XIIMIN) (FLOAT) (DEF=0.0001)
  Lower bound on the momentum fraction of the incoming parton, its value depends on the parton densities. It is important to specify this parameter since otherwise PROJET could try to evaluate the parton densities in a region where they are no longer defined, which would result in an error. XIIMIN is typically of the order of 0.0001 to 0.00001.

- ipar=2015: (IPD) (INT) (DEF=6)
  Selects the parton density distribution. If a NLO order contribution of the matrix elements is included, the parton density parametrization should be using the $\overline{\text{MS}}$-scheme.

- ipar=2016: (IPDSET) (INT) (DEF=2)
  Selects the subset of the parton density distribution.

- ipar=2017: (EPEE1) (FLOAT) (DEF=30.712)
  Defines the laboratory frame by specifying the fraction E_P/E_e in this frame. It is assumed that the incoming proton is massless. If the laboratory frame is the rest frame of a fixed target, then the event record is calculated in the following way ($\bowtie$): At first, the event record is calculated in a frame given by E_P/E_e=DABS(EPEE1). This is done under the assumption that the proton is massless. Then the event record is boosted to the fixed target frame assuming a massive proton. If the event record in the fixed target frame is needed, EPEE1 should be set to some "large" negative value, -10.0 for example.

- ipar=2018: (DTHEKPMIN) (FLOAT) (DEF=0.0)
  Cut on the minimum polar angle of the outgoing lepton in the laboratory frame (in the range $[0°, 180°]$, 0° corresponds to the proton direction).

- ipar=2019: (DTHEKPMAX) (FLOAT) (DEF=180.0)
  Cut on the maximum polar angle of the outgoing lepton in the laboratory frame (in the range $[0°, 180°]$, 0° corresponds to the proton direction).

- ipar=2020: (DTHEPIMIN) (FLOAT) (DEF=0.0)
  Cut on the minimum polar angle of the outgoing partons in the laboratory frame (in the range $[0°, 180°]$, 0° corresponds to the proton direction).

- ipar=2021: (DTHEPIMAX) (FLOAT) (DEF=180.0)
  Cut on the maximum polar angle of the outgoing partons in the laboratory frame (in the range $[0°, 180°]$, 0° corresponds to the proton direction).

- ipar=2026: (ILAMBDA) (INT) (DEF=0)
  0: use Λ_{QCD} and the order of the running coupling (1-loop, 2-loop) as specified in the parton density parametrization;
  1: use parameters #2027, #2029 instead.

- ipar=2027: (DLAMBDA) (FLOAT) (DEF=0.25)
  Λ_{QCD} for 4 flavours, used only if #2026 is set to 1.

- `ipar=2029: (IARUN) (int) (DEF=2)`
  Specifies order of running coupling, used only if #2026 is set to 1.
  1: 1-loop formula is used;
  2: 2-loop formula is used.

- `ipar=2030: (XHMI) (FLOAT) (DEF=0.0)`
  Specifies the lower bound on x_B.

- `ipar=2031: (XHMA) (FLOAT) (DEF=1.0)`
  Specifies the upper bound on x_B.

- `ipar=2032: (QQMI) (FLOAT) (DEF=0.0)`
  Specifies the lower bound on $\sqrt{Q^2}$.

- `ipar=2033: (QQMA) (FLOAT) (DEF=`$\sqrt{S_H}$`)`
  Specifies the upper bound on $\sqrt{Q^2}$.

- `ipar=2034: (YMI) (FLOAT) (DEF=0)`
  Specifies the lower bound on y.

- `ipar=2035: (YMA) (FLOAT) (DEF=1.0)`
  Specifies the upper bound on y.

- `ipar=2036: (WWMI) (FLOAT) (DEF=0.0)`
  Specifies the lower bound on $\sqrt{W^2}$.

- `ipar=2037: (WWMA) (FLOAT) (DEF=`$\sqrt{S_H}$`)`
  Specifies the upper bound on $\sqrt{W^2}$.

- `ipar=2038: (ISCALE) (INT) (DEF=0)`
  Choice of the factorisation scale. This is the scale where the parton densities are evaluated.
  0: μ_f=Q;
  1: μ_f=`RHOSCALE`$\times Q$;
  2: μ_f=`RHOSCALE`$\times W$ ($\bowtie$);
  3: μ_f=`RHOSCALE`$\times\sqrt{S_H}$ ($\bowtie$);
  4: μ_f=`RHOSCALE` $\times W^\alpha Q^\beta \sqrt{S_H y}^{1-\alpha-\beta}$ ($\bowtie$).

- `ipar=2039: (RHOSCALE) (FLOAT) (DEF=1.0)`
  Parameter `RHOSCALE` in the choice of the factorisation scale, see #2038.

- `ipar=2040: (ISCALER) (INT) (DEF=0)`
  Choice of the renormalisation scale. This is the scale where the running coupling α_s is evaluated.
  0: μ_r=Q;
  1: μ_r=`RHOR`$\times Q$;
  2: μ_r=`RHOR`$\times W$ ($\bowtie$);
  3: μ_r=`RHOR`$\times\sqrt{S_H}$ ($\bowtie$);
  4: μ_r=`RHOR` $\times W^\alpha Q^\beta \sqrt{S_H y}^{1-\alpha-\beta}$ ($\bowtie$).

- `ipar=2041: (RHOR) (FLOAT) (DEF=1.0)`
  Parameter `RHOR` in the choice of the renormalisation scale, see #2040.

- `ipar=2042: (DMFMIN) (FLOAT) (DEF=0.0)`
  ($\bowtie$) Minimal factorization scale (in GeV).

- `ipar=2043: (DMFMAX) (FLOAT) (DEF=`$\sqrt{S_H}$`)`
  ($\bowtie$) Maximal factorization scale (in GeV).

- `ipar=2044: (DMRMIN) (FLOAT) (DEF=0.0)`
  (⋈) Minimal renormalisation scale (in GeV).

- `ipar=2045: (DMRMAX) (FLOAT) (DEF=`$\sqrt{S_H}$`)`
  (⋈) Maximal renormalisation scale (in GeV).

- `ipar=2046: (ICUTTYPE) (INT) (DEF=0)`
  Specifies the invariant mass cut.
  0: $s_{ij} < cW^2$;
  1: $s_{ij} < cQ^2$ (⋈);
  2: $s_{ij} < cS_H$ (⋈);
  3: $s_{ij} < cW^\alpha Q^\beta \sqrt{S_H y}^{1-\alpha-\beta}$ (⋈).
  `ICUTTYPE`=1,2,3 is implemented for test purposes only.

- `ipar=2047: (IAEMRUN) (INT) (DEF=0)`
  Specifies whether the fine structure constant is fixed ($\alpha(0)$) or running ($\alpha(Q^2)$).
  0: fixed;
  1: running.

- `ipar=2048: (PRENA) (FLOAT) (DEF=0.5)`
  (⋈) Parameter α used in the calculation of the renormalisation scale.

- `ipar=2049: (PRENB) (FLOAT) (DEF=0.5)`
  (⋈) Parameter β used in the calculation of the renormalisation scale.

- `ipar=2050: (PFACTA) (FLOAT) (DEF=0.5)`
  (⋈) Parameter α used in the calculation of the factorization scale.

- `ipar=2051: (PFACTB) (FLOAT) (DEF=0.5)`
  (⋈) Parameter β used in the calculation of the factorization scale.

- `ipar=2052: (PCUTA) (FLOAT) (DEF=0.5)`
  (⋈) Parameter α used in the calculation of the invariant mass cut.

- `ipar=2053: (PCUTB) (FLOAT) (DEF=0.5)`
  (⋈) Parameter β used in the calculation of the invariant mass cut.

- `ipar=2054: (DMCMIN) (FLOAT) (DEF=0.0)`
  (⋈) Minimal scale in the invariant mass cut (in GeV).

- `ipar=2055: (DMCMAX) (FLOAT) (DEF=`$\sqrt{S_H}$`)`
  (⋈) Maximal scale in the invariant mass cut (in GeV).

- `ipar=2056: (ICLUSTYPE) (INT) (DEF=1)`
  1: use the mJADE algorithm;
  2: (⋈) all partons required to be at $x_F > 0$; cluster outgoing partons with the JADE cluster algorithm, do *not* include the remnant as a precluster. `ICLUSTYPE=2` is implemented for Born terms only.

- `ipar=2057: (EELMIN) (FLOAT) (DEF=0.0D0)`
  (⋈) Minimal electron energy in the laboratory frame (implemented as a cut).

- `ipar=2058: (PTPVP) (FLOAT) (DEF=0.0D0)`
  (⋈) Minimal p_T of all jets (except the remnant jet) in the proton–virtual–photon CM frame (PVP frame).

- `ipar=2059: (IFPDFLIB) (INT) (DEF=0)`
  Switch for the parton density library:
  0: `PAKPDF`,
  1: `PDFLIB`,
  2: private parametrization ($\bowtie$).

- `ipar=3002: (NPOINTS) (INT) (DEF=100000)`
  Number of points in the last `VEGAS` integration. The typical number should be of the order of 20.000 to 500.000, depending on the complexity of the process under consideration.

- `ipar=3003: (IALPH) (INT) (DEF=1)`
  Specifies whether `VEGAS` adjusts the integration grid in every iteration.
  0: integration grid fixed ($\bowtie$);
  1: adaptive integration.
  It is recommended to set `IALPH`=1.

- `ipar=3004: (NIITMX1) (INT) (DEF=10)`
  Number of iterations in the `VEGAS` grid definition run. Should be of the order of 10.

- `ipar=3005: (NPOINTS1) (INT) (DEF=100000)`
  Total number of points in the `VEGAS` grid definition run. Should be of the order of `NPOINTS` to 3*`NPOINTS`.

- `ipar=3006: (IPROGRESS) (INT) (DEF=1)`
  Flag to determine whether the progress of the integration is printed.
  0: do not print information;
  1: print information.

- `ipar=3007: (IINTPROGRESS) (INT) (DEF=1)`
  Flag to determine whether intermediate results of the adaptive integration routine are printed.
  0: do not print intermediate results;
  1: print intermediate results.

- `ipar=4001: (IXSECT(1)) (INT) (DEF=1)`
  Flag for quark initiated processes
  (1+1) jet: final state q,
  (2+1) jet: final state qg,
  (3+1) jet: final state qgg.
  0: do not include contribution;
  1: include contribution.

- `ipar=4002: (IXSECT(2)) (INT) (DEF=1)`
  Flag for gluon initiated processes ((2+1) and (3+1) jets).
  0: do not include contribution;
  1: include contribution.

- `ipar=4003: (IXSECT(3)) (INT) (DEF=1)`
  Flag for quark initiated process with no gluon in the final state ((3+1) jets only).
  0: do not include contribution;
  1: include contribution.

- `ipar=4004: (NPARTONS) (INT) (DEF=3)`
  Choice of process.
  -3: (3+1) jets, metric & longitudinal polarisations of the virtual photon;
  -2: (2+1) jets, metric, longitudinal and other polarisations of the virtual photon;
  -1: (1+1) jets, metric & longitudinal polarisations of the virtual photon;

1: (1+1) jets, all polarisations of the virtual photon;
2: (2+1) jets, all polarisations of the virtual photon;
3: (3+1) jets, all polarisations of the virtual photon.

- `ipar=4005: (NOUTFLAV) (INT) (DEF=5)`
  Number of active outgoing quark flavours. The value of this parameter should depend on Q^2 and on the available phase space in the hadronic final state.

- `ipar=4006: (IOLOOP) (INT) (DEF=1)`
  Specifies whether the Born term is included.
  0: do not include Born term;
  1: include Born term.
  0 is not recommended, since the integration tends to be unstable if the Born term is not included.

- `ipar=4007: (I1LOOP) (INT) (DEF=0)`
  ($\bowtie$) If a NLO correction is implemented:
  0: exclude virtual correction;
  1: include virtual correction.
  See also parameter #4012.

- `ipar=4008: (IREALF) (INT) (DEF=0)`
  ($\bowtie$) If a NLO correction is implemented:
  0: exclude real final state contributions;
  1: include real final state contributions.
  See also parameter #4012.

- `ipar=4009: (IREALI) (INT) (DEF=0)`
  ($\bowtie$) If a NLO correction is implemented:
  0: exclude real initial state contributions;
  1: include real initial state contributions.
  See also parameter #4012.

- `ipar=4011: (IOLOOPLT) (INT) (DEF=1)`
  In the case of `NPARTONS` < 0, this variable is a binary representation of the polarisations of the virtual photon.
  +1: metric polarisation;
  +2: longitudinal polarisation;
  +4: contribution $\sim \cos\Phi$ ((2+1) jets only);
  +8: contribution $\sim \cos 2\Phi$ ((2+1) jets only).

- `ipar=4012: (INLO) (INT) (DEF=0)`
  If a NLO correction is implemented:
  0: exclude NLO correction;
  1: include NLO correction.
  This parameter flag should be used rather than #4007, #4008, #4009. If this parameter is 0, the three mentioned parameters are set to 0, if it is 1, the three parameters are set to 1.

- `ipar=10012 (IDOINT) (INT) (DEF=1)`
  This flag determines whether the integration is performed (`IDOINT=1`) or not (`IDOINT=0`). The last option can be used for test purposes.

- `ipar=11000-11999: (user defined parameters) () (no default)`
  These flags will not be used in future versions of `PROJET`. They may be used for user defined parameters specified in an input file.

- `ipar=12001-12100:  (IUSERPAR())  (INT)  (DEF=0)`
  These are 100 integer variables for free use. They can be specified in the input file and are stored in the variable `IUSERPAR(i)`, where `ipar=12000+i`.

- `ipar=12101-12200:  (DUSERPAR())  (FLOAT)  (DEF=0.0)`
  These are 100 double precision floating point variables for free use. They can be specified in the input file and are stored in the variable `DUSERPAR(i)`, where `ipar=12100+i`.

- `ipar=12201-12300:  (CUSERPAR())  (STRING)  (DEF=' ')`
  These are 100 string variables with a length of 100 characters each for free use. They can be specified in the input file and are stored in the variable `CUSERPAR(i)`, where `ipar=12200+i`. The string is read from the standard input file (`unit=5`). This option is only useful if the parameters are read from an input file, see below. It is assumed that the string is located in the beginning of the next line which is read from the file.

The common block with user defined variables is `/userpar/`:

```
      integer iuserint,iuserfloat,iuserstring
      parameter (iuserint=100,iuserfloat=100,iuserstring=100)
      double precision duserpar
      integer iuserpar
      char*100 cuserpar
      common /userpar/
     &       duserpar(iuserfloat),
     &       iuserpar(iuserint),
     &       cuserpar(iuserstring)
```

Table 1 shows the the combinations of the parameters #4004, #4006, #4011 and #4012 for which there are cross sections implemented in `PROJET`.

In this table, the notation for the parameter #4011 is the following. To include a specific helicity, the numbers i indicated by $+i$ have to be added up. To include just the metric and longitudinal contribution in the case of (2+1) jets, the corresponding parameter value would be 3 ($=1+2$). To include all helicities, the parameter value is 15 ($=1+2+4+8$). Please note that in the case of (3+1) jets, the helicities that depend explicitly on the orientation of the leptonic and hadronic systems can not be selected separately. They are, however, included in the option "all helicities". In the case of (1+1) jet cross sections, there are no angular dependent terms, since the orientation of the leptonic and hadronic systems is specified entirely by the variable y.

Some remarks concerning the cuts are in order. The parameters #2030–#2037 are implemented by a variable transformation that transforms the complicated shape given by these cuts into a rectangle. So imposing these cuts does not lead to an inefficiency in the integration. The parameters #2018–#2021 are treated differently. The program calculates the angles of the outgoing particles in the laboratory frame and then simply sets the differential cross sections to zero if at least one of these cuts is violated.

The application of angular cuts leads to the following problem: If an acceptance cut of, for example, 10° is specified for the outgoing partons, then, for example, in the case of (1+1) jet production, only those events are counted as (1+1) jets which have an outgoing parton with an angle of more than 10° relative to the incoming proton. If the jet analysis is done with a cluster algorithm, there is a second class of events that will be classified as a (1+1) jet event. Suppose a (2+1) jet event (with all invariants of the outgoing jets larger than the jet cut) with one of the jets in the direction of 2° relative to the proton, such that this jet disappears in the beam pipe. This event in principle *is* a (2+1) jet event, but the experimental cluster algorithm would classify this as a (1+1) jet event because one of

#4004	#4006	#4011	#4012	Process
-3	1	+1	0	(3+1), Born, metric
-3	1	+2	0	(3+1), Born, longitudinal
-2	1	+1	0	(2+1), Born, metric (σ_M)
-2	1	+2	0	(2+1), Born, longitudinal (σ_L)
-2	1	+4	0	(2+1), Born, $\sim \cos\Phi$ (σ_Φ)
-2	1	+8	0	(2+1), Born, $\sim \cos 2\Phi$ ($\sigma_{2\Phi}$)
-2	1	+1	1	(2+1), Born & NLO, metric (σ_M)
-2	1	+2	1	(2+1), Born & NLO, longitudinal (σ_L)
-2	1	+4	1	(2+1), Born & NLO, $\sim \cos\Phi$ (σ_Φ)
-2	1	+8	1	(2+1), Born & NLO, $\sim \cos 2\Phi$ ($\sigma_{2\Phi}$)
-1	1	1	0	(1+1), Born, metric
-1	1	+1	1	(1+1), Born & NLO, metric
-1	1	+2	1	(1+1), Born & NLO, longitudinal
3	1	0	0	(3+1), Born, all helicities
2	1	0	0	(2+1), Born, all helicities
1	1	0	0	(1+1), Born, all helicities

Table 1: Cross sections implemented in `PROJET`.

the jets is not seen. To get a "realistic" (1+1) jet cross section, one therefore has to add the cross section of *real* (2+1) jet events that *look like* (1+1) jet events to the class of (1+1) jet events. In order to do this, the user has to use the event record and select those events that fulfill the criteria described above.

There are several other parameters not appearing in this list defined in `PROJET`. It is not recommended to use them since they are not necessarily supported in new versions of the program.

Since it is not very flexible to recompile the calling program every time when a parameter has been changed, the example program `MCEX.F` demonstrates how to use an input file `PAREX` to set all parameters. This input file is a sequential text file. The file starts with two text strings; the first gives the filename for an output file, and the second specifies a job name. Then there are consecutive lines of the form

```
ipar,par
```

Here `ipar` specifies which parameter gets the value `par`. Comments are included in the following way:

```
0,0 This is a comment...
```

Program execution continues after a line

```
-4,0
```

The input file is terminated with a line

```
-5,0
```

which terminates program execution.

7 Error Conditions

In this section a list of the error conditions that might cause a stop of `PROJET` is given. Wherever possible it has been made sure that the program cannot crash due to numerical instabilities (if this didn't result in too large a sacrifice of program efficiency). However, there are several cases in which parameters do not make sense or are not defined. In these cases, the program will print a warning or error message on the output file (standard output) and call the `user` error routine. If the error number is negative, the error is considered to be fatal, and `PROJET` execution will stop subsequently after the `user` subroutine for program termination has been called. If the error number is positive, it is just a warning, and `PROJET` continues after making a reasonable adjustment of the condition that caused the error. In several cases, the warning messages and the calls to `user` are done only a certain number of times to avoid a messy output. The following is a list of the errors that might occur:

- `ierr = +1:`
  Q^2 not valid in parton density parametrization, use maximum/minimum Q^2 as specified in the particular parametrization.
- `ierr = +2:`
  An error has been reported from `userset`.
- `ierr = +3:`
  Value of `KPAR` not valid.
- `ierr = -6:`
  Momentum fraction ξ of the incoming parton too low, parton densities no longer valid.
- `ierr = -8:`
  Form of running α_s not known.
- `ierr = -9:`
  μ_f^2 too low in running α_s.
- `ierr = +10:`
  Lepton variable y out of range.
- `ierr = -12:`
  Invariants for virtual (2+1) jet correction out of range.
- `ierr = -13:`
  Error in a call to the parton density parametrization.
- `ierr = -14:`
  Q^2, W^2 bounds not compatible with x_B, y bounds.
- `ierr = -15:`
  No space in the event record.
- `ierr = +17:`
  `dacos` function not defined, argument corrected.
- `ierr = +18:`
  `dsqrt` function not defined, argument corrected.
- `ierr = +19:`
  `dlog` function not defined, argument corrected.

- `ierr = -20:`
  `igrid` not valid.
- `ierr = -21, -22:`
  Error when `PROJET` tried to access parton density functions.

Error handling in `PROJET` needs some improvement, since it is not desirable that the program stops execution completely in case of a fatal error. This will be accomplished step by step in new versions of the program.

8 Installation

The `PROJET` package consists of

- `PROJET.TEX`: the LaTeX file of the manual;
- `PROJET.F`: a set of subroutines;
- `PROJET.MCEX.F`: an example program;
- `PROJET.PAREX`: an input file for the example program;
- `PROJET.RESEX`: the output file (`unit=6`) that `MCEX` creates;
- `PROJET.PLOTEX`: the output on an external file (`unit=55`) that `MCEX` creates;
- `PROJET.PAKPDF.F`: the library of parton density parametrizations `PAKPDF` [Cha92];
- `PROJET.PAKMAN`: the `PAKPDF` manual;
- `PROJET.PDFDUMMY.F`: a dummy routine to replace the `PDFLIB` library of parton density parametrizations (if not available).

`PROJET.F` is the set of subroutines containing the cross section formulas. The sets of subroutines `PROJET.F`, `PROJET.PAKPDF.F`, `PROJET.PDFDUMMY.F` and a main program have to be linked in order to get an executable program. A simple example for a main program is `PROJET.MCEX.F`.

In order to use the `PDFLIB` [Plo93] library of parton density functions, `PROJET.PDFDUMMY.F` has to be omitted when linking `PROJET`. Parameter #2059 is the flag that determines the choice of the parton density library.

The files are available on the DESY IBM mainframe (`IO2GAU.PROJETxx.yyy`) or by electronic mail on request. In the near future, `PROJET` will be available via `FreeHEP` on the World Wide Web as well.

9 Summary and Conclusions

`PROJET` is a parton level Monte Carlo program for the calculation of jet cross sections in deeply inelastic electron proton scattering. In the present version, the matrix elements for (1+1), (2+1) and (3+1) jet Born terms and (1+1) and (2+1) jet NLO terms for all polarisations of the exchanged virtual photon are implemented. An event record for every event is accessible by the user. This permits the calculation of event variables as well as the implementation of phase space cuts on the outgoing jets and the outgoing electron.

Please send questions, comments and suggestions to graudenz @ cernvm.cern.ch. If you want to receive updated versions of this manual, please send me your electronic mail address. I would like to encourage users to report problems with the program including a description of the problem, the parameter file and some remarks on the environment of the program (machine, operating system, `FORTRAN` compiler) to me.

Up to now, `PROJET` has been tested on several workstations (including HP, DEC and SUN workstations) and on a VAX cluster. It has not been checked whether there are any incompatibilities with `FORTRAN` on IBM mainframes.

10 Acknowledgements

I would like to thank Ch. Berger, H. Heßling, G. Ingelman, G. Kramer, N. Magnussen, C. Salgado and H. Spiesberger for interesting and stimulating discussions. I am grateful in particular to R. Nisius for discussions, his efforts to test the program and helpful suggestions for the adaptation of the program to the needs of experimental physicists. H. Spiesberger has provided me with the `FORTRAN` code for the calculation of the scale dependent fine structure constant. Support from the computer centers of CERN, DESY, LBL and the RWTH Aachen is gratefully acknowledged. This work was partly supported by a grant from the Max Kade Foundation.

References

[AEM79] G. Altarelli, R.K. Ellis, G. Martinelli, Nucl. Phys. B157 (1979) 461

[BK91] Th. Brodkorb, J.G. Körner, Z. Phys. C54 (1992) 519

[BKMS89] Th. Brodkorb, J.G. Körner, E. Mirkes, G.A. Schuler, Z. Phys. C44 (1989) 415

[BM94] Th. Brodkorb, E. Mirkes, Madison preprint MAD/PH/821, April 1994

[Cha92] K. Charchula, Comp. Phys. Comm. 69 (1992) 360

[Fel94] J. Feltesse, Talk at the Deep Inelastic Scattering workshop in Eilat, Israel, 1994, preprint DAPNIA/SPP 94-22

[GM92] D. Graudenz, N. Magnussen, in Proceedings of the HERA Workshop 1991, Vol. 2, p. 261, DESY (eds. W. Buchmüller, G. Ingelman)

[Gra90] D. Graudenz, "*Der Drei-Jet-Wirkungsquerschnitt zur Ordnung $\mathcal{O}(\alpha_s^2)$ in der tiefinelastischen Elektron-Proton-Streuung*", Desy-T-90-01, September 1990, Thesis

[Gra91] D. Graudenz, Phys. Lett. B256 (1991)

[Gra94a] D. Graudenz, Phys. Rev. D49 (1994) 3291

[Gra94b] D. Graudenz, in preparation.

[HERA] H1 Collaboration, Phys. Lett. B299 (1993) 385, Phys. Lett. B298 (1993) 469
ZEUS Collaboration, Phys. Lett. B303 (1993) 183, Phys. Lett. B306 (1993) 158

[HERA91] Proceedings of the HERA Workshop 1991, Vols. 1-3, DESY (eds. W. Buchmüller, G. Ingelman)

[KMS89] J.G. Körner, E. Mirkes, G.A. Schuler, Int. J. Mod. Phys. A4 (1989) 1781

[Lep78] G. Peter Lepage, J. Comp. Phys. 27 (1978) 192

[Lep80] G. Peter Lepage, Cornell preprint, CLNS-80/447 (1980)

[Plo93] H. Plothow-Besch, Comp. Phys. Comm. 75 (1993) 396

A Test Run

This, appendix contains the source code of the test run program `MCEX.F`, the input file `PAREX`, and the output files `RESEX` and `PLOTEX`.

A.1 Test Run Program

```
C ----------------------------------------------------------------------
C
C ---> JET MONTE CARLO PROGRAM FOR DIS
C      BASED ON PROJET
C
C ---> DIRK GRAUDENZ
C
C ---> SOURCE FILE: MCEX.F
C
C ---> 15\04\1993
C      27\08\1994
C
C ----------------------------------------------------------------------

      PROGRAM MCEX

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      CHARACTER*32 JOBN,PLOTN
      COMMON /SVAR/
     &       JOBN,PLOTN

C --- INITIALIZATION
      CALL SETPAR(-3,0.D0,IERR)
      IF (IERR .NE. 0) CALL ERRMSG(IERR)
      CALL SETPAR(-2,0.D0,IERR)
      IF (IERR .NE. 0) CALL ERRMSG(IERR)

C --- READ FILENAMES
      READ(5,*)  PLOTN
      READ(5,*)  JOBN
      WRITE(6,*) 'OUTPUT FILE:',PLOTN
      WRITE(6,*) 'JOBNAME    :',JOBN

C --- OPEN OUTPUT FILE
      OPEN(UNIT   =55,
     &     FILE   =PLOTN,
     &     ACCESS ='SEQUENTIAL',
     &     FORM   ='FORMATTED',
     &     STATUS ='UNKNOWN')

C --- READ PARAMETERS FROM EXTERNAL FILE
101   CONTINUE
         READ(5,*) ICARD,XDATA
         IDATA=IDNINT(XDATA)
         IF (ICARD .EQ. -5) GOTO 99
         IF (ICARD .EQ. -4) GOTO 102
         CALL SETPAR(ICARD,XDATA,IERR)
         IF (IERR .NE. 0) CALL ERRMSG(IERR)
      GOTO 101
```

```
 102  CONTINUE

C ------ DEFINE PROCESSES AND START INTEGRATIONS

C ------ 1+1, BORN
         CALL SETP(1,1,0,0)

C ------ 1+1, BORN & NLO, TRANSVERSE
         CALL SETP(-1,1,1,1)

C ------ 2+1, BORN
         CALL SETP(2,1,0,0)

C ------ 3+1, BORN
         CALL SETP(3,1,0,0)

C ------ 2+1, BORN, TRANSVERSE
         CALL SETP(-2,1,1,0)

C ------ 2+1, BORN & NLO, TRANSVERSE
         CALL SETP(-2,1,1,1)

C --- CLOSE OUTPUT FILE
      CLOSE(55)

99    CONTINUE
      STOP
      END

C ---------------------------------------------------------------------
C
C === SET PARAMETERS #4004, #4006, #4011, #4012
C     AND START INTEGRATION
C
C ---------------------------------------------------------------------

      SUBROUTINE SETP(I4004,I4006,I4011,I4012)

      CALL SETPAR(4004,DFLOAT(I4004),IERR)
      CALL SETPAR(4006,DFLOAT(I4006),IERR)
      CALL SETPAR(4011,DFLOAT(I4011),IERR)
      CALL SETPAR(4012,DFLOAT(I4012),IERR)
      CALL SETPAR(-1  ,DFLOAT(    0),IERR)

      RETURN
      END

C ---------------------------------------------------------------------
C
C === USER SPECIFIC SUBROUTINES
C
C ---------------------------------------------------------------------

      SUBROUTINE USER(IWHAT,IUERR1)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      INTEGER NMAXENT
      PARAMETER (NMAXENT=20)
```

```
      DOUBLE PRECISION POUT,PPOL
      INTEGER NENTRY,IPTYPE,IIDENT,INOUT
      COMMON /EVTRECORD/
     &       POUT(NMAXENT,4),
     &       PPOL(NMAXENT,4),
     &       NENTRY,IPTYPE(NMAXENT),IIDENT(NMAXENT),INOUT(NMAXENT)

      COMMON /PVPRECORD/
     &       PVP(NMAXENT,4),
     &       PVPPOL(NMAXENT,4)

      DOUBLE PRECISION EVWGT,EVXSECT,EVERROR
      INTEGER IEVACCPT
      COMMON /USERWGT/
     &       EVWGT,
     &       EVXSECT,EVERROR,
     &       IEVACCPT

      INTEGER IUSERINT,IUSERFLOAT,IUSERSTRING
      PARAMETER (IUSERINT=100,IUSERFLOAT=100,IUSERSTRING=100)
      DOUBLE PRECISION DUSERPAR
      INTEGER IUSERPAR
      CHARACTER*100 CUSERPAR
      COMMON /USERPAR/
     &       DUSERPAR(IUSERFLOAT),
     &       IUSERPAR(IUSERINT),
     &       CUSERPAR(IUSERSTRING)

      COMMON /MEMO/
     &       WGTSUM,
     &       IUERR

      IF (IWHAT .EQ. 1) THEN
C ------ PRESET
      ELSEIF (IWHAT .EQ. 2) THEN
C ------ INITIALIZATION
      ELSEIF (IWHAT .EQ. 3) THEN
C ------ BEFORE START OF INTEGRATION
         IUERR=0
         WGTSUM=0.
      ELSEIF (IWHAT .EQ. 4) THEN
         IF (IEVACCPT .EQ. 1) THEN
            WGTSUM=WGTSUM+EVWGT
         ENDIF
      ELSEIF (IWHAT .EQ. 5) THEN
C ------ FINISH
         IF (IUERR .NE. 0) THEN
            WRITE(55,*) 'CRASHED! IUERR=',IUERR
         ELSE
C --------- RESULT
            WRITE(6,*) '## INTEGRATED CROSS SECTION=', EVXSECT
            WRITE(6,*) '## ESTIMATED ERROR         =', EVERROR
            WRITE(55,*) 'XSECT =',WGTSUM
         ENDIF
      ELSEIF (IWHAT .EQ. 6) THEN
C ------ ERROR
         IF (IUERR .EQ. 0 .AND. IUERR1 .LT. 0) THEN
            IUERR=IUERR1
         ENDIF
```

```
      ENDIF

      RETURN
      END

C ---------------------------------------------------------------------

      SUBROUTINE USERSET(IPAR,IDATA,XDATA,IERR)

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

      IF (IPAR .EQ. -3) THEN
         IERR=0
         GOTO 99
      ENDIF

C --- IF CALLED, THE PARAMETER IS NOT KNOWN TO PROJET,
C     AND HENCE NOT TO MCEX.
      IERR=-1

99    CONTINUE
      RETURN
      END

C ---------------------------------------------------------------------
C
C === USER SUPPLIED CUTS
C
C ---------------------------------------------------------------------

      FUNCTION IUSERCUT()

      IMPLICIT DOUBLE PRECISION (A-H,O-Z)

C --- NO ADDITIONAL CUTS
      IUSERCUT=1

99    CONTINUE
      RETURN
      END

C ---------------------------------------------------------------------
```

A.2 Test Run Input

```
'plotex'
'jobex'
0,0 ================================================================
0,0     PARAMETER FILE FOR PROJET
0,0     MAIN PROGRAM IS MCEX.F
0,0 ================================================================
0,0 --- MODE 4: INTEGRATE OVER XB AND Q**2 WITHIN BOUNDS
2001, 4
0,0 --- DO NOT PRINT PROGRESS INFORMATION
3006, 0
3007, 0
0,0 ----------------------------------------------------------------
0,0     KINEMATICS AND CUTS
0,0 ----------------------------------------------------------------
0,0 --- CM-ENERGY
2002, 295.93
0,0 --- TYPE OF CUT (W**2 (0), Q**2 (1), SH (2))
2046, 0
0,0 --- INVARIANT MASS CUT
2008, 0.02
0,0 --- CUT ON MOMENTUM FRACTION CARRIED BY THE INCOMING PARTON
2014, 1.D-5
0,0 --- PARAMETRIZATION OF THE PARTON DENSITIES (SET/PARAM.)
2015, 6
2016, 2
0,0 --- FRACTION OF PROTON ENERGY AND ELECTRON ENERGY IN THE LAB FRAME
2017, 30.712
0,0 --- MINIMUM, MAXIMUM XH
2030, 0.0
2031, 1.0
0,0 --- MINIMUM, MAXIMUM Q
2032, 2.5
2033, 20
0,0 --- MINIMUM, MAXIMUM Y
2034, 0.0
2035, 0.5
0,0 --- MINIMUM, MAXIMUM W
2036, 0.0
2037, 50.0
0,0 ================================================================
0,0     VEGAS-PARAMETERS
0,0 ----------------------------------------------------------------
0,0 --- NUMBER OF ITERATIONS IN THE GRID-DEFINING RUN
3004, 10
0,0 --- NUMBER OF POINTS IN THE GRID-DEFINING RUN
3005, 20000
0,0 --- NUMBER OF POINTS IN THE MAIN RUN
3002, 20000
0,0 ================================================================
0,0 --- NUMBER OF ACTIVE OUTGOING FLAVOURS
4005, 5
0,0 ================================================================
0,0 --- CONTINUE WITH PROGRAM EXECUTION (I.E. PERFORM INTEGRATION)
-4,0
0,0 --- TERMINATE PROGRAM
-5,0
0,0 ================================================================
```

A.3 Test Run Output, Part 1

```
       _/_/_/_/    _/_/_/_/     _/_/_/    _/_/_/_/  _/_/_/_/ _/_/_/_/_/
      _/      _/  _/      _/  _/     _/        _/  _/           _/
     _/      _/  _/      _/  _/     _/        _/  _/           _/
    _/_/_/_/    _/_/_/_/    _/     _/        _/  _/_/_/_/     _/
   _/          _/      _/  _/     _/        _/  _/           _/
  _/          _/       _/ _/     _/  _/    _/  _/           _/
 _/          _/       _/   _/_/_/    _/_/_/   _/_/_/_/     _/


                      |
   ... THE MONTE CARLO PROGRAM FOR
                      JET CROSS SECTIONS
                       |        IN DEEPLY INELASTIC SCATTERING


      +++++++++++++++++++++++++++++++++++++++++++++++++++
      +                                                 +
      +               P R O J E T  4.1                  +
      +                                                 +
      +               AUGUST 28,  1994                  +
      +                                                 +
      +                                                 +
      +  QUESTIONS, COMMENTS AND SUGGESTIONS TO         +
      +  D. GRAUDENZ, GRAUDENZ @ CERNVM.CERN.CH         +
      +  THEORETICAL PHYSICS DIVISION, CERN             +
      +  PHONE: +41-22-767-2959, FAX: +41-22-782-3914   +
      +                                                 +
      +++++++++++++++++++++++++++++++++++++++++++++++++++


OUTPUT FILE:D/plotex
JOBNAME    :ex
 DATA CARD:        2001    0.400000D+01          4
 DATA CARD:        3006    0.000000D+00          0
 DATA CARD:        3007    0.000000D+00          0
 DATA CARD:        2002    0.295930D+03        296
 DATA CARD:        2046    0.000000D+00          0
 DATA CARD:        2008    0.200000D-01          0
 DATA CARD:        2014    0.100000D-04          0
 DATA CARD:        2015    0.600000D+01          6
 DATA CARD:        2016    0.200000D+01          2
 DATA CARD:        2017    0.307120D+02         31
 DATA CARD:        2030    0.000000D+00          0
 DATA CARD:        2031    0.100000D+01          1
 DATA CARD:        2032    0.250000D+01          3
 DATA CARD:        2033    0.200000D+02         20
 DATA CARD:        2034    0.000000D+00          0
 DATA CARD:        2035    0.500000D+00          1
 DATA CARD:        2036    0.000000D+00          0
 DATA CARD:        2037    0.500000D+02         50
 DATA CARD:        3004    0.100000D+02         10
```

```
 DATA CARD:        3005    0.200000D+05       20000
 DATA CARD:        3002    0.200000D+05       20000
 DATA CARD:        4005    0.500000D+01           5
 DATA CARD:        4004    0.100000D+01           1
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.000000D+00           0
 DATA CARD:        4012    0.000000D+00           0

 TOTAL ## OF POINTS        :     19602
 ## SURVIVING THE CUTS     :     19602
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.649022D+05
 EST. ERROR ABS. =               0.156665D+02
 EST. ERROR IN % =                 0.02413857

## INTEGRATED CROSS SECTION=    64902.203125000
## ESTIMATED ERROR         =    15.666465759277
 DATA CARD:        4004   -0.100000D+01          -1
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.100000D+01           1
 DATA CARD:        4012    0.100000D+01           1

 TOTAL ## OF POINTS        :     18522
 ## SURVIVING THE CUTS     :     18522
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.284903D+05
 EST. ERROR ABS. =               0.109468D+03
 EST. ERROR IN % =                 0.38423055

## INTEGRATED CROSS SECTION=    28490.269531250
## ESTIMATED ERROR         =    109.46831512451
 DATA CARD:        4004    0.200000D+01           2
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.000000D+00           0
 DATA CARD:        4012    0.000000D+00           0

 TOTAL ## OF POINTS        :     15552
 ## SURVIVING THE CUTS     :     14117
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.247556D+05
 EST. ERROR ABS. =               0.119762D+03
 EST. ERROR IN % =                 0.48377723

## INTEGRATED CROSS SECTION=    24755.607421875
## ESTIMATED ERROR         =    119.76199340820
 DATA CARD:        4004    0.300000D+01           3
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.000000D+00           0
 DATA CARD:        4012    0.000000D+00           0

 TOTAL ## OF POINTS        :     19683
 ## SURVIVING THE CUTS     :      9632
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.209663D+04
 EST. ERROR ABS. =               0.365540D+03
```

```
 EST. ERROR IN % =                 17.43468094

## INTEGRATED CROSS SECTION=    2096.6281738281
## ESTIMATED ERROR         =    365.54043579102
 DATA CARD:        4004   -0.200000D+01          -2
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.100000D+01           1
 DATA CARD:        4012    0.000000D+00           0

 TOTAL ## OF POINTS        :     19683
 ## SURVIVING THE CUTS     :     17623
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.191440D+05
 EST. ERROR ABS. =               0.111850D+03
 EST. ERROR IN % =                 0.58425546

## INTEGRATED CROSS SECTION=    19143.990234375
## ESTIMATED ERROR         =    111.84981536865
 DATA CARD:        4004   -0.200000D+01          -2
 DATA CARD:        4006    0.100000D+01           1
 DATA CARD:        4011    0.100000D+01           1
 DATA CARD:        4012    0.100000D+01           1

 TOTAL ## OF POINTS        :     19683
 ## SURVIVING THE CUTS     :     17849
 REQUESTED ## OF PTS/IT    :     20000

 VEGAS-INTEGRAL  =               0.209589D+05
 EST. ERROR ABS. =               0.207645D+03
 EST. ERROR IN % =                 0.99072427

## INTEGRATED CROSS SECTION=    20958.925781250
## ESTIMATED ERROR         =    207.64517211914
```

A.4 Test Run Output, Part2

```
XSECT =    64902.196744796
XSECT =    28490.199754252
XSECT =    24755.394848660
XSECT =    2096.5009340818
XSECT =    19143.702981131
XSECT =    20958.925678212
```